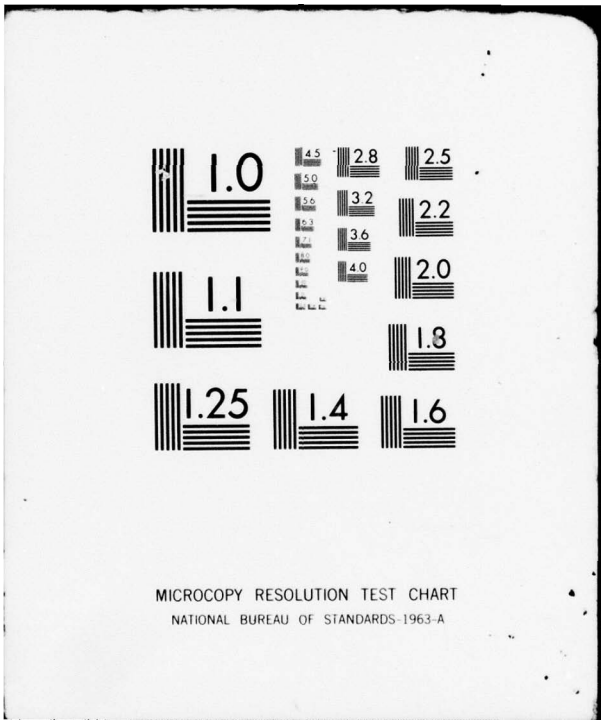
AD-A056 211 NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER WASH--ETC F/G 9/2
NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (N--ETC(U)
JUN 76

UNCLASSIFIED NMCSSC-CSM-UM-15-74-V2-2/ NL

| OF |
AD
A056211

END
DATE
FILMED
8 -78
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DEFENSE COMMUNICATIONS AGENCY

COMMAND AND CONTROL
TECHNICAL CENTER
WASHINGTON, D. C. 20301

IN REPLY
REFER TO:

NMCS Information Processing System 360
Formatted File System (NIPS 360 FFS).
Users Manual. Volume VII. Utility
Support (UT). Changes 2 and 3.

10 June 1976

62 p.

TO: DISTRIBUTION

SUBJECT: Change 2 to CSM UM 15-74, Utility Support (UT), Volume VII, dated 15 October 1976

NMCSSC-CSM-UM-15-74-V2-2/3

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.

2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.

3. When this change has been posted, make an entry in the Record of Changes on the inside cover.

FOR THE DIRECTOR:

J. DOUGLAS POTTER
Assistant to the Director
for Administration

15 Enclosures
Change 2 pages

243 000

JCB

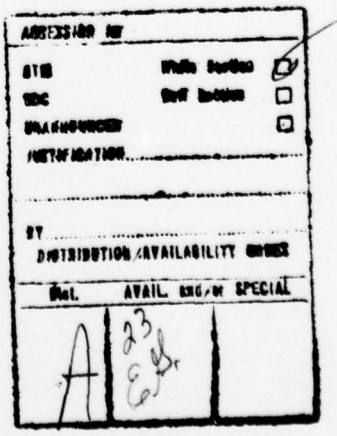
DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

EFFECTIVE PAGES - 10 June 1976

This list is used to verify the accuracy of CSM UM 15-74, Volume VII, after change 2 pages have been inserted. Original pages are indicated by the letter O, change 2 by the numeral 2.

Page No.	Change No.
Title Page	2
ii	O
iii - v	2
vi	2
1 - 14	O
15 - 18	2
19 - 52	O
53 - 56	2
57 - 76	O
76.1 - 76.2	2
77 - 81	O



COMMAND AND CONTROL TECHNICAL CENTER

Computer System Manual Number CSM UM 15-74

10 June 1976

NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (NIPS 360 FFS)

Users Manual

Volume VII - Utility Support (UT)

SUBMITTED BY:

CRAIG K. HILL
Captain USA
CCTC Project Officer

APPROVED BY:

CEDRIC S. WALLACE
Captain, USN
Deputy Director, NMCS ADP

CONTENTS

ABSTRACT

This volume defines the capabilities of NIPS 360 FFS Utility (UT) components. It describes the function of each utility, its inputs, its outputs, and serves as a reference for the knowledgeable user of these components.

This document supersedes CSM UM 15B-68, Volume VII.

CSM UM 15-74 Volume VII, is part of the following additional NIPS 360 FFS documentation:

CSM UM 15-74	Vol. I	- Introduction to File Concepts
	Vol. II	- File Structuring (FS)
	Vol. III	- File Maintenance (FM)
	Vol. IV	- Retrieval and Sort Processor (RASP)
	Vol. V	- Output Processor (OP)
	Vol. VI	- Terminal Processing (TP)
	Vol. VIII	- Job Preparation Manual
	Vol. IX	- Error Codes
TR 54-74		- Installation of NIPS 360 FFS
CSM GD 15-74		- General Description

For example, the following table value statements would supply argument/function pairs to a "Service Table":

```
E      U.S. Coast Guard
   W      U.S. Army
 J            U.S. Air Force
  N      U.S. Navy
M U.S. Marine Corps
```

The order of argument/function pairs in the example shown remains consistent in each table value statement. However, column selection of data value placement is free format. In this format the fixed length field cannot contain embedded blanks, but the variable length field can.

2.2 Job Setup

a. The following job setup is used to execute the XTABGEN cataloged procedures and must be organized in the order shown:

```
//Jobname        JOB   (Standard parameters)
//Stepname       EXEC XTABGEN,LIB=yourlib
//TAB.SYSIN      DD  *
TABLE IDENTIFICATION STATEMENT (Required)
HEADR STATEMENT                (Optional)
COMMENT STATEMENT              (Optional)
TABLE VALUE STATEMENTS         (Required)
/*
```

b. More than one table may be generated in a single job step. If tables are batched and a table argument length exceeds 30 bytes, TABGEN must be informed by stating the maximum argument length in the PARM field of the EXEC card, for example, PARM=nnn (nnn is any number between 1 and 255).

c. The generated table will become an executable load module which will reside as a member of a library that will be identified by the LIB= parameter in the EXEC card. The user may create his own new

library by overriding the following symbolic parameters:

```
LIBDISP=  (NEW,KEEP)
VLIB=     (volume)
LIBSP=    (space)
ULIB=     (unit)
```

2.3 Limitations

a. The maximum table size is approximately 528,000 bytes. The maximum number of pages which any table may contain is 132.

b. The maximum argument/function size is:

- o. 256 bytes when using keyword table identification statements
- o. 80 bytes when using fixed table identification statements.

2.4 Examples

Examples of test runs using keyword table identification statements.

a. The following TABGEN source deck setup was used to generate a fixed-to-variable length table using free format table value statements. The table name is CTRYS, and the table converts a two-character internal storage code to an output value with a maximum length of 15 characters. The user name is TABGENTEST and the table page size is 2,000 characters.

```
//Jobname       JOB   (standard parameters)
//STEPA         EXEC  XTABGEN,LIB=yourlib
//TAB.SYSIN     DD  *
TABLE=CTRYS ARG=2 FUNC=15 USE=0 PAGE=2K TABGENTEST
*    TABLE CTRYS USING FREE FORMAT TABLE
```

```
**    IDENTIFICATION CARD AND TABLE VALUE
***   CARDS.
AA  AFRICA
    AC          ATLANTIC OCEAN
.....................................................
64          CARIBBEAN
        65 PACIFIC ISLANDS
/*
```

The output listings for this source deck will will as follows:

```
DATE 71001  TABLE-CTRYS  ORIGINATOR TABGENTEST  PAGE-001

     TABLE CTRYS USING FREE FORMAT TABLE
     IDENTIFICATION CARD AND TABLE VALUE
     CARDS.

ARGUMENT  FUNCTION
--------  --------
   AA        AFRICA
   AC        ATLANTIC OCEAN
  ........................
   64        CARIBBEAN
   65        PACIFIC ISLANDS
```

b. The following TABGEN source deck setup was used to generate a variable-to-fixed length conversion table using fixed format table value statements. The table name is UNLVSI, and the table is used to convert an external value with a maximum length of 15 characters to an input storage code with a maximum length of three characters. The user name is TABGENTEST and the table page size is 1,000 bytes:

```
//Jobname       JOB  (standard parameters)
//STEPA         EXEC XTABGEN,LIB=yourlib
//TAB.SYSIN     DD  *
TABLE=UNLVSI ARG=10/24 FUNC=1/3 USE=I TABGENTEST
*     TABLE UNLVS USING KEYWORD TABLE IDENTIFICATION
**    STATEMENTS AND FIXED FORMAT TABLE VALUE
***   CARDS.
```

```
A      NUMBERED ARMY
ACD    ACADEMY
.....................................................
U      UNIT
USS    US SHIP
WG     WING
/*
```

The output listing for this source deck will appear as follows:

```
DATE 71007  TABLE-UNLVSI  ORIGINATOR-TABGENTEST  PAGE-001

     TABLE UNLVS USING KEYWORD TABLE IDENTIFICATION
     STATEMENT AND FIXED FORMAT TABLE VALUE
     CARDS.

ARGUMENT          FUNCTION
--------          --------
NUMBERED ARMY     A
ACADEMY           ACD
.....................................
UNIT              U
US SHIP           USS
WING              WG
```

Table 1 (optional) - Name of Stop Word Table

Table 2 (optional) - Name of Dictionary

Sub 1 (optional) - Name of User Scan Routine

TEXT — Hyphen Option
Default is TEXT

DROP
RETAIN
SEPARATE

TEXT - Always retain as part of text value.

DROP - Always drop from value. Superfluous if (1) text characters immediately precede and follow hyphen or (2) immediately preceded by a text character and followed by one or more blanks and a text character. If neither of these cases apply, hyphen is treated as a word separator.

RETAIN - Same rules that apply to DROP, except that when the hyphen is not a word separator it is kept in the value as a text character.

SEPARATE - Hyphen is always treated as a word separator.

The delete operation requires only the first three operands - INDEX, the field name, and the action keyword, DELETE. For a secondary indexed field, the add operation requires only the first two operands -- INDEX and the field name -- but may optionally provide for a conversion and/or analyzer subroutine or table. In this case, the action parameter ADD must be present.

If only one subroutine or table is designated, its function - as a conversion subroutine or as an analyzer subroutine - will be determined from the parameters that are

specified in the SUB/TAB statement defining the subroutine. If both parameters are present, the first must be the conversion subroutine, and the second must be the analyzer subroutine. An analyzer subroutine or table may not be used as a conversion routine, and a conversion subroutine or table may not be used as an analyzer subroutine in the same Index Specification run.

For a keyword indexed field, the action parameter KEYWORD must be present after the ADD parameter in order to differentiate the two types of indexed fields. Any combination of stop word table, dictionary scan routine or hyphen option may be specified in any order.

Example of an Add Index Action:

```
INDEX  MEQPT  ADD  CONSUB
```

Example of a Delete Index Action:

```
INDEX CNTRY DELETE
```

11.2 UTNDXSPC Output

UTNDXSPC builds and inserts Index Descriptor Records into the data file for indexes added and deletes Index Descriptor Records for indexes deleted. It then calls Index Maintenance to either generate or update the Index Data Set. UTNDXSPC also lists a summary of actions performed plus any error conditions encountered.

11.3 UTNDXSPC Job Setup

The following JCL cards are used to invoke the cataloged procedure XSP which will either generate or update a disk-resident Index Data Set based on the ISAM or SAM data file.

```
//JOBNAME   JCB     (Standard Parameters)
#//STEPNAME  EXEC   XSP,PARM=GEN,ISAM=filename,
#//                 VISAM='SER=xxxxxx',
//                  XVOL='SER=yyyyyy',
```

```
//                YDISP=(aaaa),
//                BLKSIZE=zzzz,
//                NBRBLK=nnn,
//                XINDEX=indxname,
//                LAB=bb,DEN=c,VSMOUT='SER=dddddd',SAMOUT=samname
/*
```

where:

#PARM=GEN - Required only when creating an Index Data Set from a file already containing Index Descriptor records in the FFT.

filename - Name of ISAM or SAM data file.

xxxxxx - Volume serial number of the disk-resident ISAM data file or tape-resident SAM data file.

yyyyyy - Volume serial number of the disk-resident Index Data Set

aaaa - Required only when generating a new Index Data Set

zzzz - Blocksize, needed only when generating a new Index Data Set and overriding the default size. Blocksize must be between 560 and 1020 bytes.

nnn - Number of blocks, needed only when generating a new Index Data Set and overriding the default size. A minimum of 50 blocks must be allocated.

indxname - Name of Index Data Set. It must be the same as the SAM or ISAM operand. The name will be suffixed with an X by the procedure.

The following parameters are required only when processing a SAM data base:

bb	-	Label processing
c	-	Tape processing
dddddd	-	Volume serial number of the output SAM data file
samname	-	Name of output SAM data file. (Omitted if same as input file.)

#Section 16

Format Definition Translator Utility (UTODE)

UTODE is used tc place format definitions on a user library. A format definition gives a description of a CRT display format and the Input Message Queue records to be created from data entered on the display. Before a user can call for the display format at a CRT terminal, the format definition must reside on the user library. UTODE creates skeleton and IMQ table control blocks from the format definition source statements and writes the control blccks into the user library using the display format name as the member name.

16.1 Input

Input to UTODE will consist of one or more format definitions. The format definition source statements are described in Section 6 (FORMATTER) of the Terminal Processing Users Manual. The input source statements may be in punched cards or in card image records stored in a partitioned data set.

16.2 Job Setup

The following JCL statements illustrate the deck setup used to execute UTODE. In the first setup, the input definition source statements are in punched cards.

```
//AA      EXEC XUTODE,LIB=TEST360,VLIB='SER=MYPACK'
//SYSIN DD *

          Definition source statement cards.

/*
```

The following JCL would be used if the input were card image records stored in a library.

```
//AB       EXEC AUTODE,LIB=TEST360,VLIB='SER=MYPACK'
//SYSIN  DD DSN=MYLIB(FORMAT1),VOL=SER=MYPACK2,
//           DISP=(SHR,KEEP),UNIT=2314,
//           DCB=(RECFM=FM,LRECL=80,BLKSIZE=800)
/*
```

DEFENSE COMMUNICATIONS AGENCY
COMMAND AND CONTROL
TECHINCAL CENTER
WASHINGTON, D.C. 20301

IN REPLY
REFER TO:

15 October 1977

TO: DISTRIBUTION

SUBJECT: Change 3 to CSM UM 15-74, Utility Support (UT), Volume VII, dated 15 October 1974

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.

2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.

3. When this change has been posted, make an entry in the Record of Changes on the inside cover.

FOR THE DIRECTOR

46 Enclosures
Change 3 pages

J. DOUGLAS POTTER
Assistant to the Director
for Administration

EFFECTIVE PAGES - 30 September 1977

This list is used to verify the accuracy of CSM UM 15-74, Volume VII, after change 3 pages have been inserted. Original pages are indicated by the letter O, change 3 by the numeral 3.

Page No.	Change No.
Title Page	O
ii	O
iii-iv	3
v-vi	O
1-2	3
3-24	O
25-32	3
32.1-32.2	3
33-38	O
39-43	3
43.1-43.2	3
44	3
45-48	O
49-50	3
51-52	O
53-62	3
62.1-62.2	3
63-68	3
68.1-68.2	3
69-76	O
76.1-76.2	3

CONTENTS

Section 1

INTRODUCTION

This volume of the User Manual contains an analytical description of the general utility support functions provided by NIPS 360 FFS. These functions perform common processing required by all the components. The purpose, use, deck setup, and options of each capability are presented along with clarifying examples.

This volume is divided into the following sections:

a. TABGEN - Discusses the user conversion table generator function

b. SUBIDR - Discusses the user conversion subroutine loader function

c. Data Conversion - Discusses the data base conversion function which converts a 1410 FFS data base to an equivalent NIPS 360 FFS data base

\# d. File Load/Unload - Provides the Job Control Language to transfer files to and from the Sequential Access Method and the Indexed Sequential Access Method or the Virtual Storage Access Method.

e. UTQRTQDF - Discusses the creation of a NIPS 360 FFS data file from the answer file (QRT/QDF) produced by the retrieval processor (RASP)

f. UTSUBCHK - Discusses the user subroutine checkout function

g. UTDMPLIB - Discusses the capability of printing the names of reports and/or logic statements currently residing on the data file

h. UTCLASS - Discusses the capability of changing the classification of a data file.

i. SOURCE LANGUAGE STORAGE - Discusses the library storage of source programs to facilitate housekeeping and program maintenance.

j. UTNDXSPC - Discusses the manner in which indexing information may be used without running an FS or FI job.

k. UTCLASS - Discusses the capability which permits the user to transfer the entire data set, from one resident medium to the other.

l. UTFLDSCN - Discusses the source statement field reference scan function.

where

aaaaaaa = NIPS 360 FFS ISAM data base name on disk
bbbbbb = disk serial number for the ISAM data base
ccccc = 1410 FFS tape data base name
dddddd = tape serial number for the 1410 data base
eeeeeee = NIPS 360 FFS tape data base name
ffffff = tape serial number for the 360 data base.

4.1.2 Procedure X1410CON

This procedure will convert a NIPS 360 FFS data base to a 1410 FFS data base. The NIPS 360 FFS data base is assumed to be on a 2314 disk pack and the converted data base is written on a seven-track tape with nonstandard labels. Using symbolic parameters for this procedure, one would use the following JCL:

```
//JOBNAME     JCB    (standard parameters)
//       EXEC  X1410CON,ISAM=aaaaaaa,VISAM='SER=bbbbbb',    X
//       SAM=eeeee,VSAM='SER=ffffff'
//SYSIN  DD   *
                    ----
                (1410 FFS FFT object deck)
                    ----
/*
```

where

aaaaaaa = NIPS 360 FFS ISAM data base name on disk
bbbbbb = disk serial number for the ISAM data base
eeeee = data base name for the 1410 FFS tape
ffffff = tape serial number for the 1410 data base.

Section 5

FILE LOAD/UNLOAD UTILITIES

The following paragraphs provide the job control language and necessary information required to transfer files to and from the Sequential Access and Indexed Sequential Access Methods (SAM and ISAM) and to and from the Sequential Access and Virtual Storage Access Methods (SAM and VSAM). These utilities are effectively automatic and require only the JCL stream for control.

5.1 SAM to ISAM or VSAM Utility - UTBIDISM

This utility will build an NIPS ISAM data base or a NIPS VSAM data base from a NIPS SAM data base.

Also, under control of the CC PARM on the EXEC card it can optionally build the ISAM or VSAM data base in the compressed and/or compacted form or reverse the process to produce the ISAM or VSAM data base in the standard form.

The JCL DD cards used are:

//DATAFILE	DD	Parameters defining the new ISAM data base
//SAMFILE	DD	Parameters defining the existing SAM data base
//VSMFILE	DD	Parameters defining the new VSAM data base.

The new ISAM NIPS data base will be created using the following parameters: write check, master index, cyl/def

overflow, independent overflow, delete option and feedback reorganization criteria.

The procedure XSTCIS is used to build a NIPS 360 FFS ISAM or VSAM data base from a NIPS 360 FFS SAM data base. It is assumed that the existing NIPS SAM data base resides on a nine-track tape, and that the ISAM data base will reside on a 2314 disk pack with a disposition of KEEP. If output is to be a VSAM data base, it must have been previously defined via the VSAM service routine IDCAMS and cataloged on a NIPS user catalog. Using symbolic parameters for this procedure, one would use the following JCL:

```
//JOBNAME      JCB    (standard parameters)
//             EXEC   XSTOIS,ISAM=aaaaaaa,VISAM='SER=bbbbbb', X
//    PRIME=cc,INDEX=d,SAM=eeeeeee,VSAM='SER=ffffff',
#   //     PARM='ggggggg'
    /*
```

where:

aaaaaaa	=	name used for the new ISAM data base
bbbbbbb	=	disk serial number for the new ISAM data base
cc	=	number of cylinders of prime space for the new ISAM data base
d	=	size of index needed for the new ISAM data base
eeeeeeee	=	name of the existing SAM data base
ffffff	=	serial number of the tape that contains the existing SAM data base.
# ggggggg	=	parameter option for data record transformation (COMPRESS, COMPACT, EXPAND or both COMPRESS and COMPACT separated by a comma) or, for SAM to VSAM, replace the ISAM parameter with:

```
VSCAT=aaaaaaaa,VSDSN=bbbbbbbb,NONVSM='DUMMY,'
```

where:

aaaaaaaa	=	name of the NIPS user catalog
# bbbbbbbb	=	name of the VSAM data base to be loaded.

Keywords in the PARM are used to designate the form of the output ISAM or VSAM data base irrespective of the form of the input SAM data base. The keywords for the PARM and their implications are as follows:

COMPRESS - compression is applied to the output data records.

COMPACT - compaction is applied to the output data records

COMPRESS, COMPACT - a combination of compression and compaction is applied to the output data records

EXPAND - compression and/or compaction is reversed to produce standard form data records.

#If none of the parameters is specified, no data transformation takes place, i.e., the output ISAM or VSAM data base form will be the same as the input SAM data base form.

Normally, after the file has been copied to disk, the remaining prime space will be filled with pad records. These are ISAM records with valid keys which have the delete byte set on. These records make it possible to add records to the end of the file during later update and still remain in the prime area.

If the user does not want these pad records, he may include the NOPAD keyword in the PARM on the EXEC card.

An example might be a SAM file that is dumped to disk for use with TP. In this case, it would be desirable not to pass all of the pad records for a query, while the user is waiting at the terminal.

Upon completion of UTBLDISM, the user will receive either the message,

'NIPS ISAM (VSAM) DATA FILE HAS BEEN SUCCESSFULLY CREATED'

if the job was successful, or a USER 0200 ABEND if the job was unsuccessful.

#5.2 ISAM or VSAM to SAM Utility - UTBIDSAM

This utility will build a SAM data base from either an ISAM, a VSAM or another SAM data base. Under control of the PARM on the EXEC card, it can optionally build the output SAM data base in the compressed and/or compacted form. Also, prior to termination of the utility, it will print statistics that are beneficial for the user/analyst.

The first line of statistics printed gives the file name, volume serial number, date, time, and page number. Then information is printed concerning the sets.

5.3 Compression and Compaction of Data Records

Compression and compaction provide a means for the reduction of intermediate storage requirements for data without altering the integrity of the data. This data reduction scheme is particularly suited to data files that contain strings of identical characters or a large quantity of alphabetic data.

A string of identical characters is compressed by translating it to two bytes. The first byte is a control byte which indicates that compression has been applied and gives a count of the number of identical consecutive bytes that were in the original string. The second byte is identical to those in the original string.

A string of alphabetic characters is compacted by translating it to a control byte followed by a string of coded characters. The control byte indicates that compaction has been applied and gives a count of the coded characters. Each coded character represents a combination of two adjacent alphabetic characters.

There are seven columns of information displayed, with headings, as follows:

a.	SET	The first entry is "FIXED SET" and following in each periodic set if it the number specified in the FFT.
b.	MINIMUM SUBSET SIZE (BYTE)	Shows the size, in bytes, of the smallest subset within the specified set, fixed or periodic. If a variable field is specified, the minimum and maximum size will not be the same.
c.	MAXIMUM SUBSET	Shows the size, in bytes, of the largest subset within the specified set, fixed or periodic.

	SIZE (BYTES)	If a variable field is specified, the minimum and maximum size will not be the same.
d.	NUMBER OF SUBSETS PER DATA RECORD MINIMUM MAXIMUM	These two columns reflect the minimum and maximum number of subsets within a periodic for the total record. The fixed set would be 1 for minimum and maximum. The absolute minimum for periodics would be Ø and the maximum, any variable number.
e.	TOTAL NUMBER OF SUBSETS	The total number of subsets in each periodic set for the entire file is printed here. The number printed for the fixed set is the total number of data records in the file.
f.	SIZE OF LARGEST SET	This field shows the maximum size in bytes of the indicated set for any record in the file.

For the ISAM file, the above-mentioned statistics are printed along with information on the organization of the ISAM file.

The number of PRIME, OVERFLOW, DELETE, and PAD records is calculated and printed. Also information from the DSCB is printed. The DSCB information concerns the INDEX, PRIME, and OVERFLOW cylinder/track allocation and usage. The column headings are "CYL/TRACKS ALLOCATED," "TRACKS UTILIZED," and "PERCENT TRACKS UTILIZED." The number of tracks in each cylinder overflow area is also provided. With this information the user can calculate the amount of space needed for his job. This alleviates unnecessary pad records that are added if any prime area remains.

The JCL DD cards used are:

```
//DATAFILE DD   parameters defining existing ISAM data file

#//VSMFILE  DD  parameters define existing VSAM data file

//SAMFILE  DD   parameters defining existing SAM data file
```

```
//SAMOUT   DD   parameters defining new SAM data file
```

The procedure XISTOS is used to build a NIPS 360 FPS SAM data file from an ISAM or VSAM data file or copy a NIPS 360 SAM data file. It is assumed that the existing ISAM data file resides on a 2314 disk pack or that the existing SAM data file resides on a nine-track tape with standard label. If input is a VSAM data file, it must be cataloged on a NIPS user catalog. It is also assumed that the output SAM data file will be written on a standard label nine-track tape.

Using symbolic parameters for this procedure, one would use the following JCL:

```
//JOBNAME     JOB(Standard parameters)
//           EXEC XISTOS,ISAM=aaaaaa,VISAM='SER=bbbbbb',
//           SAM=eeeeee,VSAM='SER=ffffff',PARM=gggggg
/*
```

where:

aaaaaa = name of the existing ISAM data file

bbbbbb = serial number of the disk volume where the ISAM data file resides

eeeeee = name used for the new SAM data file

ffffff = serial number for the new SAM data file

gggggg = parameter option for data record transformation (COMPRESS, COMPACT, EXPAND or both COMPRESS and COMPACT separated by a comma).

or:

```
//JOBNAME    JOB        (Standard parameters)
//           EXEC XISTOS,OLDSAM=aaaaaa,OLDVSAM='SER=bbbbbb',
//           SAM=eeeeee,VSAM='SER=ffffff',PARM=gggggg
```

where:

aaaaaa = name of the existing SAM data file

bbbbbb = serial number of the volume where the SAM data file resides

eeeeee = name used for the new SAM data file

ffffff = serial number for the new SAM data file

gggggg = parameter option for data record transformation (COMPRESS, COMPACT, EXPAND or both COMPRESS and COMPACT separated by a comma) or, for VSAM to SAM, replace the ISAM parameters with:

VSCAT=aaaaaaaa,VSDSN=bbbbbbbb

where:

aaaaaaaa = name of NIPS user catalog
# bbbbbbbb = name of exisitng VSAM data file.

Keywords in the PARM are used to designate the form of the output SAM data base irrespective of the form of the input ISAM, SAM or VSAM data base. The keywords for the PARM and their implications are the same as those for the SAM to ISAM utility.

Upon successful completion of UTBLDSAM, the user will receive the message:

'NIPS SAM DATA FILE HAS BEEN SUCCESSFULLY CREATED'.

cards are required for each argument - use blank cards as filler if necessary).

7.3 Job Setup

The following statements illustrate the job setup used to execute the XSUBCHK procedure:

```
//      EXEC   XSUBCHK,LIB=TESTER
//SUBCHK.SYSIN  DD *

(Control card defining input as described in 7.1)

data card(s)
/*
```

The LIB symbolic parameter defines the partitioned data set where the subroutine's executable load module is stored. The library need not be a NIPS file library, but it should have the same DCB attributes.

Sample Control Card

```
MYSUB              C99
```

Sample data cards in the above case

```
                                        Col 80
123456789CAQFT86789ABCB ..... 123796

789056789 BCBDEFGHQ

1379654321EDLC86789DFQC ..... 569324

8432976843AFQDETGHQ
```

Section 8

UTDMPLIB

The utility program UTDMPLIB allows the user/analyst to print the names of reports and/or logic statements associated with a NIPS 360 FFS data file. The output from this utility is a formatted listing. The input data file may be in Sequential Access Method (SAM), Indexed Sequential Access Method (ISAM) or Virtual Storage Access Method (VSAM) form.

8.1 Input

Input for the cataloged procedure XDMPLIB consists of the user's file (SAM, ISAM or VSAM) and a single control card. The control card is in a fixed format in that it must begin in card column 1 and parameters must be separated by commas. The control card is to be prepared as follows:

```
PRINT,REPORT,XXXXXXX
```

where XXXXXXX will contain one of the following parameters:

ALL	- provides a listing of all report names and all logic statements.
LIST	- provides a list of all report names.
(Report ID)	- provides a listing of all logic statement names for the report ID specified.

8.2 Job Setup

The follcwing statements will assist to illustrate the job setup required to execute the XDMPLIB procedure:

```
//       EXEC   XDMPLIB,ISAM=TESTER
//UTDMP.SYSIN   DD  *
PRINT,REPORT,ALL
/*
```

The preceding example will list all report and statement names on a cataloged ISAM data set named TESTER.

```
#//  EXEC  XDMPLIB,VSCAT='NIPS.CAT',ISAM='VSAM.TESTER'
//UTDMP.DATAFILE  DD  AMP='AMORG'
//UTDMP.SYSIN  DD  *
PRINT,REPORT,ALL
/*
```

The preceding example will list all report and statement names on the VSAM file VSAM.TESTER cataloged on NIPS.CAT.

```
//       EXEC  XDMPLIB,ISAM=TESTER,VISAM='SER=FFSLIB'
//UTDMP.SYSIN     DD  *
PRINT,REPORT,REPTA
/*
```

This example will list associated logic statement names for a report entitled REPTA included in the uncataloged ISAM data set TESTER residing on a 2314 volume labeled FFSLIB.

```
//       EXEC  XDMPLIB,SAM=TESTER,VSAM='SER=N12345'
//UTDMP.SYSIN     DD  *
PRINT,REPORT,LIST
/*
```

The preceding example provides a listing of all report names contained within the data set TESTER which resides on a nine-track, 800 bpi tape with the volume serial number of N12345.

The XDMPIIB procedure may be executed in a separate job or as a step of, for example, an FM update. It is to be noted that all routines used in conjunction with the execution of this procedure reside on FFS.JOBLIB.

Section 9

UTCLASS

The program UTCLASS provides the user with the capability of changing the classification of a NIPS 360 FFS ISAM, SAM or VSAM file. A single card is required which will contain only the new classification of the file.

9.1 Input

The program UTCLASS requires a single input card containing the new classification of the data file. The card is free format and the new classification will be left-justified with trailing blanks, if required, when written out to the file. The length of the classification field is 32 characters and truncation will be performed to the right.

To change classification of the file to blanks, that is, no classification, the input card must contain at least one blank enclosed in apostrophes. This is the only case in which an apostrophe which is entered on the input card does not become part of the actual file classification.

9.2 Output

Two outputs are produced by a successful run of this program:

a. The updated data file containing the new classification.

b. A listing on the printer indicating successful or unsuccessful updating and the new classification if successful.

```
//CLASS.SYSIN  DD  *
     (user-supplied card ccntaining new classification)
/*
```

The preceding example will change the classification of the VSAM file, VSAM.TESTER, catalcged on NIPS.CAT.

9.3 Job Setup

The following statements illustrate the deck setup used to execute the UTCLASS procedure.

```
//        EXEC   XCLASS,ISAM=TESTER,
//        VISAM='SER=FFSLIB'
//CLASS.SYSIN DD  *

     (User-supplied card containing new
     classification)

/*
```

The preceding examples will charge the classification on an uncataloged ISAM file.

```
//     EXEC XCLASS,SAM=TESTER,VSMOUT='SER=TSTVOL',
//          SAMOUT=

     (User-supplied card containing new
     classification)

/*
```

The preceding example will change the classification of a cataloged SAM file on tape.

```
//     EXEC   XCLASS,SAM=TESTER,
//          USAM=2314,VSAM='SER=FFSLIB'
//CLASS.SYSIN DD *

     (User-supplied card containing new
     classification)

/*
```

The preceding example will change the classification on an uncataloged SAM file on disk.

```
#//  EXEC  XCLASS,VSCAT='NIPS.CAT',ISAM='VSAM.TESTER'
//CLASS.DATAFILE  DD  AMP='AMORG'
```

Section 10

SOURCE LANGUAGE STORAGE

Source programs for NIPS components may be stored on a library to facilitate housekeeping and program maintenance. Maintaining a library will ensure that a single, current version of the source program is always available. The Source Program Library can be updated by batch jobs, or by the EDIT component if on-line terminals and the NIPS TP component are available. EDIT capabilities are described in Volume VI of the NIPS User Manuals, Terminal Processing (TP).

10.1 Source Libraries

Two types of libraries can be utilized for storage. Both types are direct-access partitioned data sets. The first type of library is normally used to store 80-character card images. This library is similar to a Procedure or Macro Library and has fixed length, 80-character records, normally blocked to an efficient blocksize which is a multiple of 80.

The second type of library is the File Library. This library has records of undefined length and is normally used to store the compiled, executable NIPS user programs (retrievals, RITs, etc.). Source programs will be stored on this library in 800-character blocks, containing 10 card images each. OS utilities may not be used to modify source on this type of library, but NIPS components may as discussed below.

Each source program member stored on a library by a NIPS language component or utility will contain an indicator as to which NIPS language component (RASP, FM, etc.) is involved. This will enable a terminal user to scan a library for RITs, logic statements, etc.

Section 11

INDEX SPECIFIER (UTNDXSPC)

The Index Specification utility allows a user to specify indexing information for a data file without running a File Structure or File Maintenance job. The user can add and delete indexes in the same run. For each index added, all information necessary to make that index operational is generated and placed in the Index Data Set.

Any fixed-length field defined in the file may be specified as a secondary index. Any variable field, variable set or fixed-length alpha field defined in the file may be specified as a keyword index. A fixed-length field may not be specified as both a secondary and keyword index.

UTNDXSPC acts as the driver to perform the functions of calling Index Specification to insert or delete Index Descriptor Records in the data file, and of executing Index Maintenance to correlate the Index Descriptor Records in the data file with the Index Control Records in the Index Data Set and to update the latter accordingly. UTNDXSPC operates on an ISAM, SAM or VSAM data file.

11.1 UTNDXSPC Input

UTNDXSPC accepts SUB/TAB and INDEX statements as input. These statements must be submitted through the SYSIN device. Further discussion of Index Specification may be found in Volume II, File Structuring.

11.1.1 SUB/TAB Card

A SUB/TAB card is used to describe a subroutine or table to be used by secondary indexing. A subroutine or table may be a conversion routine, to convert data from an internal data file format to a separate index format. On the other

hand, a subroutine or table may be an analyzer routine, designed to analyze the parameter list of a FUNCTION operator to determine index usage and provide a list of values for index qualification.

For a keyword indexed field, a subroutine or table defines options that direct the selection of keyword values. It may designate a stop word table which is a list of irrelevant (noise) words, a dictionary which is a list of all non-literal keywords, or a user scan routine, which is a subroutine provided by the user to process the keyword indexed fields.

One SUB/TAB card must be submitted for each subroutine referred to by the INDEX statements. A subroutine or table may be defined for one function only in an Index Specification run, and each unique subroutine or table name may be submitted only once per run.

All SUB/TAB statements must appear first in the input stream, before the INDEX statements.

The SUB/TAB statement is free-format. The operands must be in order, each separated from the others by at least one blank. A period is used after the last entry to signify the end of the statement.

Operand	Meaning
SUB SUBROUTINE TAB TABLE	Statement identifier; may be any one of these.
Subname	Name of subroutine or table. This name must follow the conventions for system names as described in Volume 1, Introduction to File Concepts.

Table 1 (optional) - Name of Stop Word Table

Table 2 (optional) - Name of Dictionary

Sub 1 (optional) - Name of User Scan Routine

TEXT
DROP
RETAIN
SEPARATE

Hyphen Option
Default is TEXT

TEXT - Always retain as part of text value.

DROP - Always drop from value. Superfluous if (1) text characters immediately precede and follow hyphen or (2) immediately preceded by a text character and followed by one or more blanks and a text character. If neither of these cases apply, hyphen is treated as a word separator.

RETAIN - Same rules that apply to DROP, except that when the hyphen is not a word separator it is kept in the value as a text character.

SEPARATE - Hyphen is always treated as a word separator.

The delete operation requires only the first three operands - INDEX, the field name, and the action keyword, DELETE. For a secondary indexed field, the add operation requires only the first two operands -- INDEX and the field name -- but may optionally provide for a conversion and/or analyzer subroutine or table. In this case, the action parameter ADD must be present.

If only one subroutine or table is designated, its function - as a conversion subroutine or as an analyzer subroutine - will be determined from the parameters that are

specified in the SUB/TAB statement defining the subroutine. If both parameters are present, the first must be the conversion subroutine, and the second must be the analyzer subroutine. An analyzer subroutine or table may not be used as a conversion routine, and a conversion subroutine or table may not be used as an analyzer subroutine in the same Index Specification run.

For a keyword indexed field, the action parameter KEYWORD must be present after the ADD parameter in order to differentiate the two types of indexed fields. Any combination of stop word table, dictionary scan routine or hyphen option may be specified in any order.

Example of an Add Index Action:

```
INDEX  MEQPT  ADD  CONSUB
```

Example of a Delete Index Action:

```
INDEX CNTRY DELETE
```

11.2 UTNDXSPC Output

UTNDXSPC builds and inserts Index Descriptor Records into the data file for indexes added and deletes Index Descriptor Records for indexes deleted. It then calls Index Maintenance to either generate or update the Index Data Set. UTNDXSPC also lists a summary of actions performed plus any error conditions encountered.

11.3 UTNDXSPC Job Setup

The following JCL cards are used to invoke the cataloged procedure XSP which will either generate or update a disk-resident Index Data Set based on the ISAM, SAM or VSAM data file.

```
//JOBNAME   JCB    (Standard Parameters)
//STEPNAME  EXEC   XSP,PARM=GEN,ISAM=filename,
//                 VISAM='SER=xxxxxx',
```

```
//                  XVOL='SER=yyyyyy',
//                  XDISP=(aaaa),
//                  BLKSIZE=zzzz,
//                  NBRBLK=nnn,
//                  XINDEX=indxname,
#//                 VSCAT=nipcat,
//                  LAB=bb,DEN=c,VSMOUT='SER=dddddd',SAMOUT=samname
#//UTXSP.NEWFILE  DD  AMP='AMORG'
/*
```

where:

PARM=GEN - Required only when creating an Index Data Set from a file already containing Index Descriptor records in the FFT.

#filename - Name of ISAM, SAM or VSAM data file.

xxxxxx - Volume serial number of the disk-resident ISAM data file or tape-resident SAM data file.

yyyyyy - Volume serial number of the disk-resident Index Data Set

aaaa - Required only when generating a new Index Data Set

zzzz - Blocksize, needed only when generating a new Index Data Set and overriding the default size. Blocksize must be between 560 and 1020 bytes.

nnn - Number of blocks, needed only when generating a new Index Data Set and overriding the default size. A minimum of 50 blocks must be allocated.

indxname - Name of Index Data Set. It must be the same as the SAM or ISAM operand. The name will be suffixed with an X by the procedure.

The follcwing parameters are required only when processing a SAM data base:

bb - Iabel processing

c - Iape processing

dddddd - Volume serial number of the output SAM data file

samname - Name cf output SAM data file. (Omitted if same as input file.)

The follcwing parameter and the NEWFILE DD override statement are used only when processing a VSAM data file:

#nipcat - Name cf the NIPS user catalog for the VSAM data file.

Section 12

Index Transfer (UTNDXTFR)

An Index Data Set may reside on a direct-access device or on tape, but only the disk-resident medium can be used by any NIPS component. Index Transfer (UTNDXTFR) will permit the user to transfer the entire data set, from one resident medium to the other.

The primary use of UTNDXTFR is to reorganize the disk-resident indexes. Initially, a disk-resident Index Data Set is packed with index information. As the indexes are maintained, gaps or unused areas may occur in the data set as records are deleted and others are added. By using UTNDXTFR, the user can transfer the disk-resident data (only the valid information is transferred) to tape, and again from tape back to disk. This operation condenses the data set. The tape so created may be retained as a backup.

In the disk to tape mode of operation, a statistical printout of unique values (for secondary indexed fields) and keywords (for keyword indexed fields) and their occurrences are optionally developed. Binary values will be converted to decimal for ease of reading. However, any dictionary fields using conversion subroutines or keyword fields having synonyms in the dictionary, will be printed in the converted form, just as they appear in the Index Data Set.

12.1 UTNDXTFR Input

Two cataloged procedures are available for invoking the UTNDXTFR utility. XTRDISK will transfer a disk-resident Index Data Set to a sequential access medium, while XTRTAPE will reconstruct an Index Data Set from a previously unloaded tape version of the disk data set.

12.2 UTNDXTFR Output

UTNDXTFR produces an Index Data Set on the residence medium indicated.

12.3 UTNDXTFR Job Setup

The following JCL cards are used to invoke the cataloged procedure XTRDISK which will unload a disk-resident Index Data Set to a sequential access medium:

```
//JOBNAME   JOB     (Standard Parameters)
//STEPNAME  EXEC    XTRDISK,XFNAME=AAAAAAAA,
//                  XFVCL=BBBBBB,XTNAME=CCCCCCCC,
//                  XTVOL=DDDDDD,
//                  STAT=YES,(optional)
//                  ISAM=EEEEEEE,VISAM=FFFFFF
#//                 VSCAT=XXXXXXXX,
#//                 SAM=GGGGGG,VSAM=HHHHHH
#//XTR.DATAFILE  DD  AMP='AMORG'
/*
```

where:

AAAAAAAA	-	Name of the Index Data Set (the user must supply the "X" suffix)
BBBBBB	-	Volume serial number of the disk-resident Index Data Set
CCCCCCCC	-	Name of the unloaded version of the Index Data Set
DDDDDD	-	Volume serial number of the sequential access device which will contain the unloaded Index Data Set
optional parameters	-	required only when the statics option (STAT=YES) is chosen.
#EEEEEEE	-	Name of the ISAM or VSAM data file

FFFFFF - Volume serial number of the ISAM data file

GGGGGGG - Name of the SAM data file

HHHHHHH - Volume serial number of the SAM data file

#XXXXXXX - Name of the NIPS user catalog for the VSAM data file.

The DATAFILE DD override statement is used only for processing VSAM data files.

The following JCL cards are used to invoke the cataloged procedure XTRTAPE which will reconstruct a disk-resident Index Data Set from a previously unloaded sequential version of the Index Data Set:

```
//JOBNAME   JCE    (Standard Parameters)
//STEPNAME  EXEC   XTRTAPE,XTNAME=AAAAAAAA,
//                 XTVOL=BBBBBB,XFNAME=CCCCCCCC,
//                 XFVOL=DDDDDD
```

where:

AAAAAAAA - Name of the disk-resident Index Data Set (user must supply the "X" suffix)

BBBBBB - Volume serial number of the disk-resident Index Data Set

CCCCCCCC - Name given to the unloaded version of the Index Data Set

DDDDDD - Volume serial number of the device containing the unloaded version of the Index Data Set

Section 13

UTFLDSCN

UTFLDSCN will scan the NIPS components source statements and provide the user with the ccunt of data fields referenced in the source statements. This utility is useful in helping the analyst to determine the activity of his data fields. This will assist the user in determining which fields are candidates for index fields in the Secondary Indexing capability.

The utility will prccess source statements pertaining to a single file in one execution. Multifile RITs and multifile queries will be accepted as input; however, only the data fields cf the input data file will be processed. All other files will be ignored. In order tc completely process multi-file RITs and multifile queries, it would be necessary to include the source statements in an execution for each file referenced.

UTFLDSCN will output a listing of the scurce input statements fcllcwed by a listing cf the count of references for the data fields and a summary listing of data field reference court for each batch compcnent. A transaction record will also be output for each data field referenced in a single source input statement.

13.1 UTFIDSCN Input

Input to the utility will consist of a NIPS data file in SAM, ISAM or VSAM fcrmat, a ccntrcl card, and the source input statements and/or members of a partitioned data set.

The format cf the input contrcl card is as follows:

```
./ SOURCE  COMP=XXXX,NAME=SNAME,MEMBER=MNAME
```

where ./ must be in cclumns 1 and 2 fcllowed by one or blanks.

COMP=XXXX where XXXX may be FM, RASP, OP, or CUIP to identify the component.

NAME=SNAME where SNAME is the name of the source statement. If the source is a logic statement, the name must be the report name and logic statement names enclosed in quotes. If the report name is less than seven characters, it must be padded with blanks to seven characters.

MEMBER=MNAME where MNAME is the member name of the source statement on a partitioned data set. If this operand is used, a partitioned data set must be included in the job stream. If this operand is omitted, the source statements must follow the control card in the input stream.

13.2 UTFLDSCN Output

The outputs from UTFLDSCN are as follows:

Source Listing - The source input records in input order.

Field Listing - This output will consist of a header for each source deck indicating file name, component name, source name, and member name, if any. The body of the listing will consist of only those fields referenced and the count of references. This will follow the source listing for each source module. After all source modules have been processed, a summary listing will be provided containing the count of the field references per component.

Transaction Data Set - The transaction data set will be 50 characters long with an 'S' in column one to be used as a logic statement name. The format is as follows:

```
Column  1    - CHARACTER 'S'
       2-8   - FILE NAME
       9-12  - COMPONENT NAME
      13-25  - SOURCE MODULE NAME
      26-33  - FIELD NAME
      34-36  - SET NUMBER, DECIMAL
      37-42  - COUNT OF REFERENCES, DECIMAL
      43-48  - DATE - MMDDYY
      49-50  - UNUSED
```

Parameters may be entered in the PARM field on the EXEC card to suppress output. They are as follows:

NS - Suppress printing of source input

NI - Suppress field and summary listing

NT - Suppress transaction output.

13.3 Job Setup

The utility will be executed by a procedure, XUTFSCAN. If the input statements are members of a partitioned data set, the data set must be included in the job stream. The following statements illustrate the job setup used to execute the XUTFSCAN procedure against the ISAM data file TEST360:

```
//  EXEC XUTFSCAN,ISAM=TEST360,LIB=TEST360
//SYSIN  DD  *
./   SOURCE  COMP=FM,NAME='TESTO  A',MEMBER=TESTA
./   SOURCE  COMP=BASE,NAME=TEST01
      BASP  Query Statements
        :
        :
./  SOURCE  COMP=OP,NAME=TESTFFF
     OP RIT Statements
       :
./  SOURCE  COMP=QUIP,NAME=QTEST,MEMBER=QTESTA
/*
```

```
#   The following JCL could be used to execute the  XUTFSCAN
procedure against the VSAM data file VSAM.TEST360:

#//  EXEC  XUTFSCAN,ISAM='VSAM.TEST360',LIB=TEST360,
//        VSCAT='NIPS.CAT'
//UTFLDSCN.DATAFILE  DD  AMP='AMORG'
//UTFLDSCN.SYSIN  DD  *
          (field scan control statements)
/*
```

Section 14

UTNDXKAN

UTNDXKAN provides the user with the capability to analyze the words in fields for which keyword indexing is to be specified. The results of the analysis can be used to determine the contents of stop word tables and dictionaries that are to be associated with these fields. If the system scan subroutine does not recover words as the user desires, a user-written scan subroutine can be used with this utility. All the words contained in a data base field can be displayed, or those words which are irrelevant (noise words) can be selected and compared to the words in the system stop word table. If that table is not adequate, the Dictionary Maintenance utility can be used to build a user stop word table. Then this utility is able to list all relevant (nonstop) words which can be used to determine dictionary requirements. If a dictionary is not employed, the words not in the stop word table will all appear as keyword entries in the index data set. Through dictionary application, synonomous words and words with varying suffixes can be collected under one Index Data Set entry, and the synonym or suffixed form can still be used in a query statement. The Dictionary Maintenance utility can be used to build a dictionary so that this utility can produce a list of the words which will become Index Data Set entries together with the keywords (including synonyms and suffixed words) associated with them which can be used as query arguments.

UTNDXKAN processes either a SAM, an ISAM or a VSAM data file. The fields that are to be processed are specified by control statements. For each field specified, the utility obtains from the control statement or from the file itself the names of the scan subroutine, stop word table and dictionary required to process that field. It scans all values in the data base (unless the number of records to be processed has been limited by the user), optionally matches the recovered words to a stop word table, then optionally matches the remaining words to a dictionary. The actual

functions performed are controlled by accepting names to FFT entry specifications or by specifying BYPASS. Override names or BYPASS must be specified for all functions if the FFT entry does not indicate keyword indexing.

UTNDXKAN displays word lists with record frequency counts or, optionally, record identifications for each field processed or for all fields as a group. Frequency count reflects the number of NIPS records in which a word appears at least once.

14.1 Input

UTNDXKAN accepts FILE and FIELD control statements. The FILE statement is optional. At least one field statement is required. Control statements are coded on cards or as card images and are contained in columns 1 through 71. Each statement must begin in a new record with the statement identifier FILE= or FIELD= in column 1 of that record. The file or field name shall immediately follow the statement identifier. A statement that exceeds 71 characters can be continued on one or more additional cards in columns 1 through 71. A nonblank character must be placed on column 72 to indicate continuation. A control statement can be interrupted after any comma or blank. Words may not be split between records. Column 72 of the last or only record of each control statement must be blank. Columns 73-80 of all records are ignored.

A control statement operand is made up of two or more keyword parameters. Each operand must be preceded and followed by one or more blanks or commas unless an operand terminates in column 71, in which case a continuation character (in column 72) may follow the operand. If multiple values are specified for an operand, at least one blank or comma must separate each value and the group of values must be enclosed in parentheses. The operands can be coded in any order. No extra commas are required to indicate omitted operands.

14.1.1 FILE Statement

The FILE statement may be omitted. It applies to file processing and affects all the fields to be processed. It may appear only once and must be the first statement. Its operands allow control of file access and output merging.

The statement identifier is:

FILE=filename - name of file. Must be coded in column 1.

The statement operands are:

MERGE=YES - merge the word lists from all fields into one group of lists that reflects the entire file.

BYPASS=one or more bypass options - valid only if MERGE=YES is specified. It defines the word lists to be suppressed for all fields and overrides the FIELD statement BYPASS operand. The display lists identified by the following terms are omitted from the output for all fields:

STOP	- stop word table matches
NONKEY	- stop word table and dictionary non-matches
KEYWORD	- dictionary matches including synonym sublists
SYNONYM	- keyword sublists
SUFFIXES	- dictionary nonmatches which are composed of keywords with valid suffixes; the keywords appear in the keyword list.

SKIP=nnnnn - skip nnnnn NIPS logical records before processing any fields, where nnnnn is a number between 1 and 32767.

STOPAFT-nnnnn - stop after processing nnnnn NIPS logical records, where nnnnn is a number between 1 and 32767.

14.1.2 FIELD Statement

The FIELD statement identifies a field to be analyzed. At least one FIELD statement is required and up to 50 are allowed.

The statement identifier is:

FIELD=fieldname - name of field or variable set. Must be coded in column 1.

The statement operands are:

SCAN=name of scan subroutine - identifies the scan subroutine to be used instead of the FFT scan subroutine or the system scan subroutine.

STOP=name of stop word table or BYPASS - overrides the FFT stop word table specification or specifies BYPASS to cause the stop word table match function to be omitted for this field in which case all recovered words are nonstop words.

DICT-name of dictionary or BYPASS - overrides the FFT dictionary specifications or specifies BYPASS to cause the dictionary match function to be omitted for this field, in which case all nonstop words are keywords.

HYPHEN=hyphen option - overrides the FFT hyphen option specification. Valid options are TEXT, DROP, RETAIN and SEPARATE.

BYPASS=one or more bypass options - not applicable if BYPASS was specified on a FILE statement. It defines the word lists to be suppressed for the field. The display lists identified by the following terms are omitted from the output for the field:

STOP - stop word table matches

NONKEY - stop word table and dictionary non-matches

KEYWORD - dictionary matches including synonym sublists

SYNONYM - keyword sublists

SUFFIXES - dictionary nonmatches which are composed of keywords with valid suffixes; the keywords appear in the keyword list.

RECID=YES - specifies that record identifications are to be shown in word lists instead of frequency counts.

14.2 Output

UTNDXKAN displays one list of words with either frequency counts or major record identifications for each field processed or for all fields as one group if the merge option is specified. A two-character code associated with each word identifies its type. Words from types for which bypass is specified are omitted from the list. If a dictionary was specified and a data word matched a convert synonym or was suffixed, the dictionary word which will be substituted for the data word is inserted after the data word at an offset.

14.3 Job Setup

The following JCL statements illustrate the deck setup used to invoke the XKA cataloged procedure for a cataloged ISAM file and a cataloged user library:

```
//  EXEC XKA,ISAM=filename,LIB=libname
//XKA.SYSIN DD *
        (user-supplied control statements)
/*
```

where

filename - name of the NIPS ISAM data file

libname - name of library containing user scan subroutines, stop word tables and dictionaries.

The following JCL statements illustrate the deck setup used to invoke the XKA cataloged procedure for an uncataloged SAM file and an uncatalcged user library:

```
//  EXEC XKA,SAM=filename,VSAM='SER=aaaaaa',
//    LIB=libname,VLIE='SER'bbbbbb'
//XKA.SYSIN DD *
       (user- supplied control statements)
/*
```

where

filename - name of the NIPS SAM data file

aaaaaa - serial number of the SAM data file

libname - name of the library containing user scan subroutines, stop word tables, and dictionaries

bbbbbb - serial number of the user library.

The follcwing JCL statements illustrate the deck setup used tc invoke the XKA cataloged procedure for a VSAM data file cataloged on a NIPS user catalog:

```
#   //  EXEC XKA,VSCAT=nipcat,
    //   ISAM=filename,LIB=libname
    //XKA.DATAFILE  DD  AMP='AMORG'
    //XKA.SYSIN  DD  *
             (user-supplied control statements)
    /*
```

where:

#	nipcat	- name of NIPS user catalog for the VSAM data file
#	filename	- name of the NIPS VSAM data file
#	libname	- name of the library containing user scan subrcutines, stop word tables, and dictionaries.

Section 16

Format Definition Translator Utility (UTODE)

UTODE is used to place format definitions on a user library. A format definition gives a description of a CRT display format and the Input Message Queue records to be created from data entered on the display. Before a user can call for the display format at a CRT terminal, the format definition must reside on the user library. UTODE creates skeleton and IMQ table control blocks from the format definition source statements and writes the control blocks into the user library using the display format name as the member name.

16.1 Input

Input to UTODE will consist of one or more format definitions. The format definition source statements are described in Section 6 (FORMATTER) of the Terminal Processing Users Manual. The input source statements may be in punched cards or in card image records stored in a partitioned data set.

16.2 Job Setup

The following JCL statements illustrate the deck setup used to execute UTODE. In the first setup, the input definition source statements are in punched cards.

```
//AA      EXEC XUTODE,LIB=TEST360,VLIB='SER=MYPACK'
//SYSIN DD *

        Definition source statement cards.

/*
```

The following JCL would be used if the input were card image records stored in a library.

```
//AB       EXEC XUTODE,LIB=TEST360,VLIB='SER=MYPACK'
//SYSIN DD DSN=MYLIB(FORMAT1),VCL=SER=MYPACK2,
//           DISP=(SHR,KEEP),UNIT=2314,
#    //         DCB=(RECFB=FB,LRECL=80,BLKSIZE=800)
     /*
```